NETCONF Server and RESTCONF Server Configuration Models

draft-ietf-netconf-server-model-07

NETCONF WG

IETF 93 Prague

Updates since IETF 92

17 changes made since Last Call on March 3rd

Edits are in, issues are now in REVIEW state

Please review edits made for issues that interest you!

- Replaced "application" with "NETCONF/RESTCONF client" (issue #32).
- Reverted back to YANG 1.1 if-feature statements (issue #34).
- Removed import by revisions (issue #36).
- Removed groupings only used once (issue #37).
- Removed upper-bound on hello-timeout, idle-timeout, and max-sessions (issue #38).
- Clarified that when no listen address is configured, the NETCONF/RESTCONF server will listen on all addresses (issue #41).
- Update keep-alive reference to new section in Call Home draft(issue #42).
- Modified connection-type/persistent/keep-alives/interval-secs default value, removed the connection-type/periodic/linger-secs node, and also removed the reconnect-strategy/interval-secs node (issue #43).
- Clarified how last-connected reconnection type should work across reboots (issue #44).
- Clarified how DNS-expanded hostnames should be processed (issue #45).
- Removed text on how to implement keep-alives (now in the call-home draft) and removed the keep-alive configuration for listen connections (issue #46).
- Clarified text for .../periodic-connection/timeout-mins (issue #47).
- Fixed description on the "trusted-ca-certs" leaf-list (issue #48).
- Added optional keychain-based solution in appendix A (issue #49).
- Fixed description text for the interval-secs leaf (issue #50).
- moved idle-time into the listen, persistent, and periodic subtrees (issue #51).
- put presence statements on containers where it makes sense (issue #53).

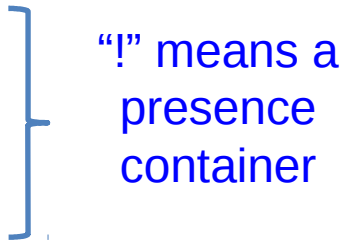
Two issues remain open

Issue #53

Goal: Put presence statements where possible

So far added for the persistent and periodic containers

```
+--rw call-home {(ssh-call-home or tls-call-home)}?
|  +--rw netconf-client* [name]
|     +--rw connection-type
|     |  +--rw (connection-type)?
|     |     +--:(persistent-connection)
|     |     |  +--rw persistent!
|     |     |     +--rw idle-timeout?   uint32
|     |     |     +--rw keep-alives
|     |     |        +--rw max-wait?       uint16
|     |     |        +--rw max-attempts?   uint8
|     |     +--:(periodic-connection)
|     |        +--rw periodic!
|     |           +--rw idle-timeout?        uint16
|     |           +--rw reconnect_timeout?   uint16
```



But what about the “listen” container?

A default for netconf-ssh (port 830) would be nice...

- but how to fake an endpoint “name”?

```
+--rw netconf-server
   +--rw listen {(ssh-listen or tls-listen)}?
      +--rw max-sessions?   uint16
      +--rw idle-timeout?   uint16
      +--rw endpoint* [name]
         +--rw name    string
         +--rw (transport)
            +--:(ssh) {ssh-listen}?
            |  +--rw ssh
            |     +--rw address?     inet:ip-address
            |     +--rw port?        inet:port-number
            |     +--rw host-keys
            |        +--rw host-key*   string
            +--:(tls) {tls-listen}?
               +--rw tls
                  +--rw address?        inet:ip-address
                  +--rw port?           inet:port-number
                  +--rw certificates
                     +--rw certificate*   string
```

What name to use?

Issue #49

- The **ietf-netconf-server** model duplicates data
  - trusted-ca-certs
  - trusted-client-certs

Same data in both the /ssh and /tls subtrees

- Making it worse, the data is duplicated again in the **ietf-restconf-server** model.

- Data duplication is bad.

Four Options

- **Solution SM49-01**
  - Do nothing [Issue --> DEAD]
  - This solution does not solve the problem.

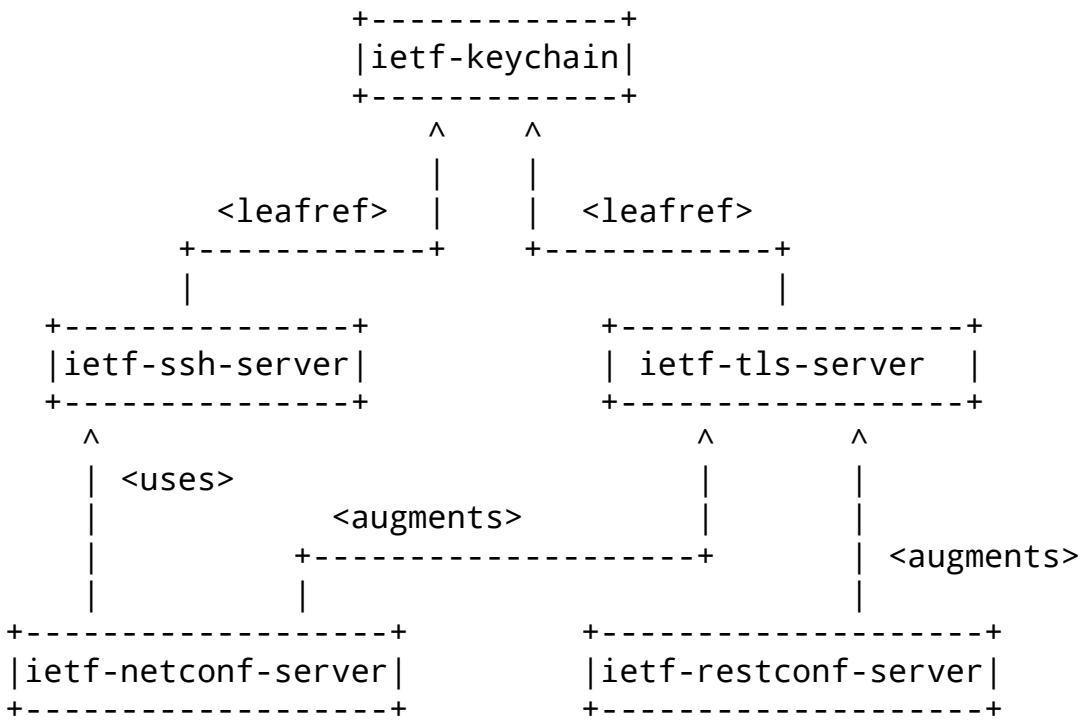
- **Solution SM49-02**
  - Unify the data structures in ietf-netconf-server only.
  - Better, but still duplication across the two YANG models: ietf-netconf-server and ietf-restconf-server

Four Options (cont.)

- **Solution SM49-03**
  - Unify the data structures in both ietf-netconf-server and ietf-restconf-server, using a 3rd YANG module defined in the server-model draft.
  - Solves the issue, but misses the opportunity to have an even better keychain based solution.

- **Solution SM49-04**
  - Define a keychain based solution.
  - Best solution, but would delay server-model draft some.

A Keychain-based model (Appendix A)



The ietf-keychain Module

```
module: ietf-keychain
   +--rw keychain
      +--rw private-keys
      |  +--rw private-key* [name]
      |     +--rw name            string
      |     +--ro algorithm?      enumeration
      |     +--ro key-length?     uint32
      |     +--ro public-key?     string
      |     +--rw certificates
      |        +--rw certificate* [name]
      |           +--rw name     string
      |           +--rw chain?   binary
      +--rw trusted-certificates* [name]
         +--rw name                    string
         +--rw trusted-certificate* [name]
            +--rw name           string
            +--rw certificate?   binary
rpcs:
   +---x generate-private-key
   |  +---w input
   |     +---w name          string
   |     +---w algorithm     enumeration
   |     +---w key-length    uint32
   +---x generate-certificate-signing-request
      +---w input
      |  +---w private-key?   -> /keychain/private-keys/private-key/name
      |  +---w subject        binary
      |  +---w attributes?    binary
      +--ro output
         +--ro certificate-signing-request    binary
```

Some fields read-only

Certificates can be registered

Keychain Example

```
<keychain xmlns="urn:ietf:params:xml:ns:yang:ietf-keychain">

    <!-- private keys and associated certificates -->
    <private-keys>
      <private-key>
        <name>TPM key</name>
        <algorithm>rsa</algorithm>
        <key-length>2048</key-length>
        <public-key>...</public-key>
        <certificates>
          <certificate>
            <name>IDevID Certificate</name>
            <chain>...</chain>
          </certificate>
        </certificates>
      </private-key>
    </private-keys>

    <!-- trusted netconf/restconf client certificates -->
    <trusted-certificates>
      <name>
         Trusted certificates for
         netconf/restconf client
      </name>
      <trusted-certificate>
        <name>George Jetson</name>
        <certificate>...</certificate>
      </trusted-certificate>
      <trusted-certificate>
        <name>Fred Flinstone</name>
        <certificate>...</certificate>
      </trusted-certificate>
    </trusted-certificates>


    // CONTINUED AT RIGHT  --->
```

```
    // CONTINUATION FROM LEFT

    <!-- trust anchors for netconf/restconf clients -->
    <trusted-certificates>
      <name>Trust anchors for netconf/restconf clients</name>
      <trusted-certificate>
        <name>Example.com</name>
        <certificate>...</certificate>
      </trusted-certificate>
    </trusted-certificates>

    <!-- trust anchors for random HTTPS servers on Internet --
    <trusted-certificates>
      <name>Trust anchors for random HTTPS servers</name>
      <trusted-certificate>
        <name>Example.com</name>
        <certificate>...</certificate>
      </trusted-certificate>
    </trusted-certificates>

  </keychain>
```

The ietf-ssh-server Module

Fake container around grouping used to generate tree diagram

```
module: ietf-ssh-server
   +--rw fake-ssh-server
      +--rw host-keys
      |  +--rw host-key* [name]
      |     +--rw name           string
      |     +--rw (type)?
      |        +--:(public-key)
      |        |  +--rw public-key?    -> /kc:keychain/private-keys/private-key/name
      |        +--:(certificate)
      |           +--rw certificate?   -> /kc:keychain/private-keys/private-key/\
      |                                   certificates/certificate/name {ssh-x509-
certs}?
      +--rw client-cert-auth {ssh-x509-certs}?
         +--rw trusted-ca-certs?       -> /kc:keychain/trusted-certificates/name
         +--rw trusted-client-certs?   -> /kc:keychain/trusted-certificates/name
```

Leafrefs

SSH Server Example

```
<fake-ssh-server xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server">
    <host-keys>
      <host-key>
        <name>IDevID</name>
        <certificate>
          IDevID Certificate
        </certificate>
      </host-key>
    </host-keys>
     <client-cert-auth>
      <trusted-ca-certs>
        Trusted certificates for netconf/restconf clients
      </trusted-ca-certs>
      <trusted-client-certs>
        Trust anchors for netconf/restconf clients
      </trusted-client-certs>
    </client-cert-auth>
  </fake-ssh-server>
```

leafref

leafref

leafref

The ietf-tls-server Module

Fake container around grouping used to generate tree diagram

```
module: ietf-tls-server
   +--rw fake-tls-server
      +--rw certificates
      |  +--rw certificate* [name]
      |     +--rw name    -> /kc:keychain/private-keys/private-key/certificates/certificate/name
      +--rw client-auth
         +--rw trusted-ca-certs?       -> /kc:keychain/trusted-certificates/name
         +--rw trusted-client-certs?   -> /kc:keychain/trusted-certificates/name
```

Leafrefs

TLS Server Example

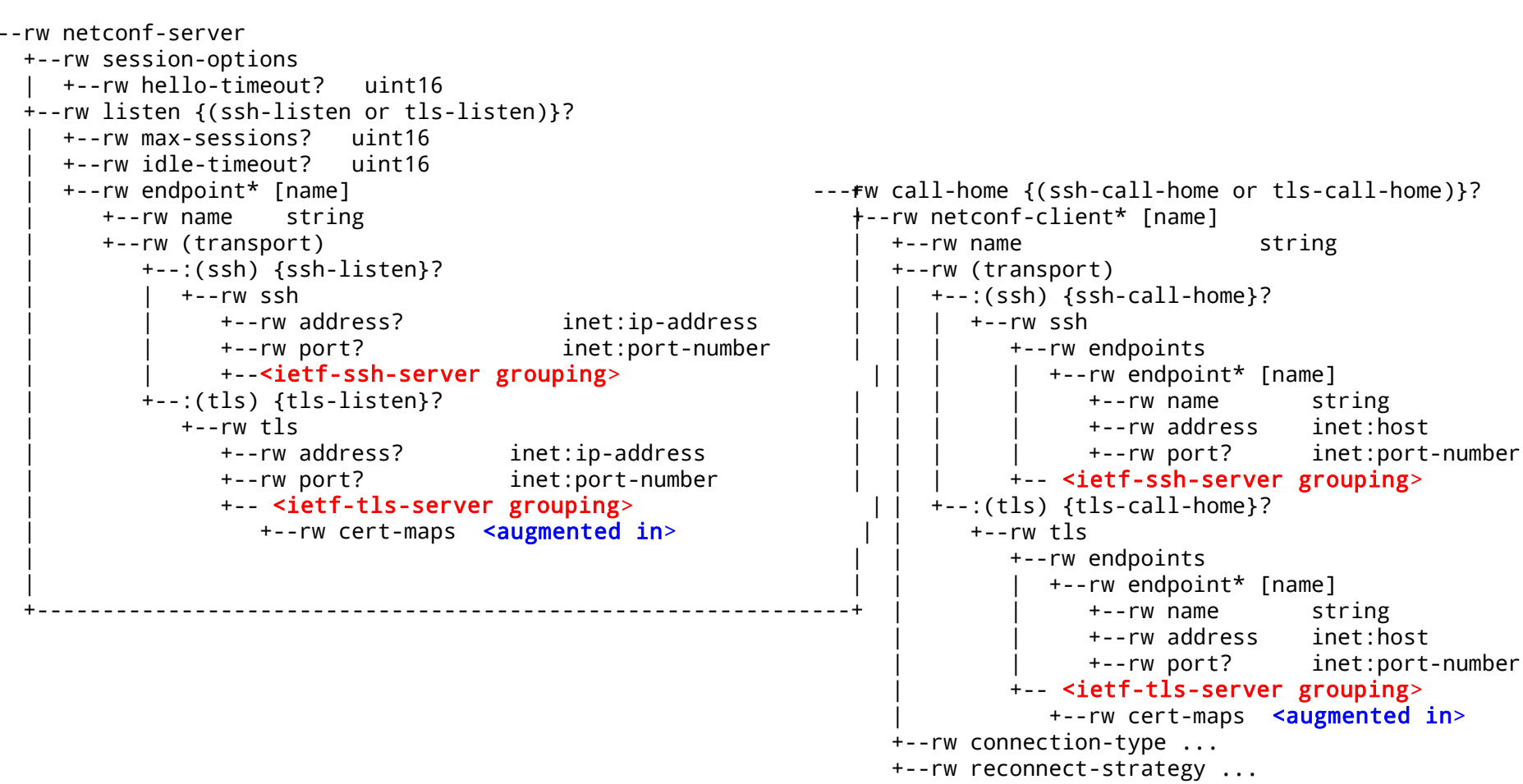
```
<fake-tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
    </certificates>
      <certificate>
        IDevID Certificate
      </certificate>
    </certificates>
    <client-auth>
      <trusted-ca-certs>
        Trusted certificates for netconf/restconf clients
      </trusted-ca-certs>
      <trusted-client-certs>
        Trust anchors for netconf/restconf clients
      </trusted-client-certs>
    </client-auth>
  </fake-tls-server>
```

leafref

leafref

leafref

The ietf-netconf-server Module



NETCONF Server Example (SSH only)

```
<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <endpoint>
      <name>netconf/ssh</name>
      <ssh>
        <address>11.22.33.44</address>
         <ietf-ssh-server data goes here>
      </ssh>
    </endpoint>
  </listen>
 <call-home>
    <netconf-client>
      <name>config-mgr</name>
      <ssh>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>11.22.33.44</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>55.66.77.88</address>
          </endpoint>
        </endpoints>
         <ietf-ssh-server data goes here>
      </ssh>
    </netconf-client>
  </call-home>
</netconf-server>
```

The ietf-restconf-server Module

```
module: ietf-restconf-server-new
   +--rw restconf-server
      +--rw listen {tls-listen}?
      |  +--rw max-sessions?   uint16
      |  +--rw endpoint* [name]
      |     +--rw name    string
      |     +--rw (transport)
      |        +--:(tls)
      |           +--rw tls
      |              +--rw address?        inet:ip-address
      |              +--rw port?           inet:port-number
      |              +-- <ietf-tls-server grouping>
      |                 +--rw cert-maps  <augmented in>
      +--rw call-home {tls-call-home}?
         +--rw restconf-client* [name]
            +--rw name                  string
            +--rw (transport)
            |  +--:(tls)
            |     +--rw tls
            |        +--rw endpoints
            |        |  +--rw endpoint* [name]
            |        |     +--rw name       string
            |        |     +--rw address    inet:host
            |        |     +--rw port?      inet:port-number
            |        +-- <ietf-tls-server grouping>
            |           +--rw cert-maps  <augmented in>
            +--rw connection-type ...
            +--rw reconnect-strategy ...
```

RESTCONF Server Example

```
<restconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <endpoint>
      <name>netconf/ssh</name>
      <ssh>
        <address>11.22.33.44</address>
        <ietf-tls-server data goes here>
          <plus additional cert-maps data from augmentation>
      </ssh>
    </endpoint>
  </listen>
 <call-home>
    <netconf-client>
      <name>config-mgr</name>
      <ssh>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>11.22.33.44</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>55.66.77.88</address>
          </endpoint>
        </endpoints>
        <ietf-tls-server data goes here>
          <plus additional cert-maps data from augmentation>
      </ssh>
    </netconf-client>
  </call-home>
</restconf-server>
```

Comments on Keychain model?

Thank You